Lab 5: Directions

Student : Abdullah Bohamad

Instructor: Dr Goncalo Martins

Date of Experiment: Friday, 17 / October / 2025 - Friday, 24 / October / 2025

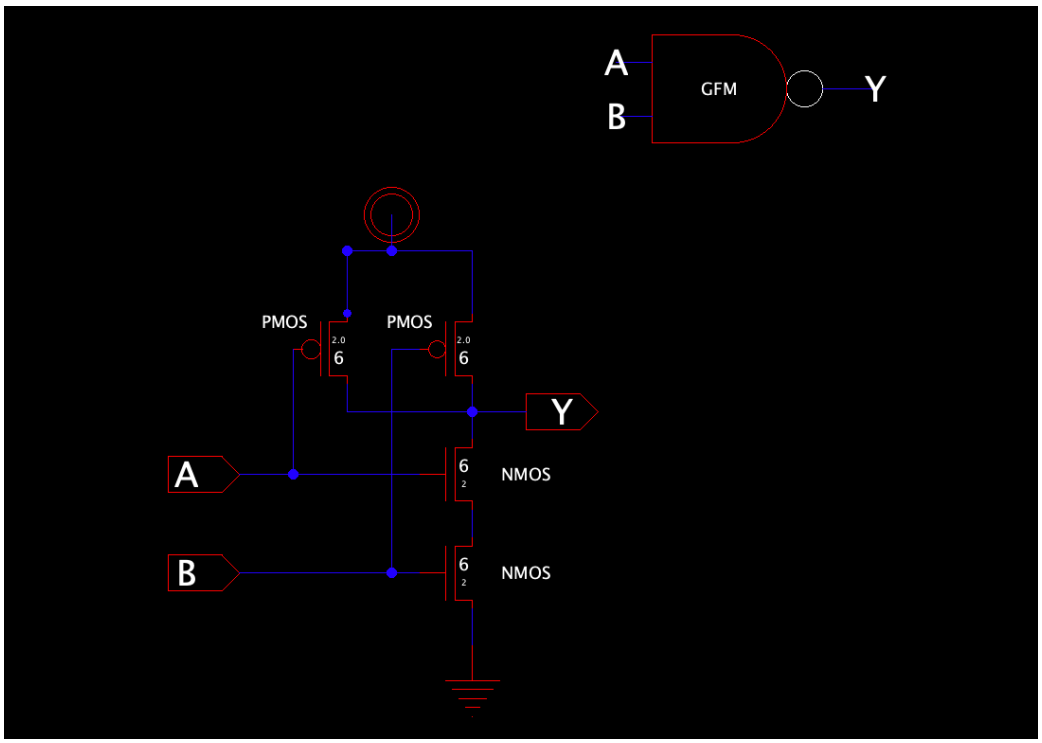
Introduction:

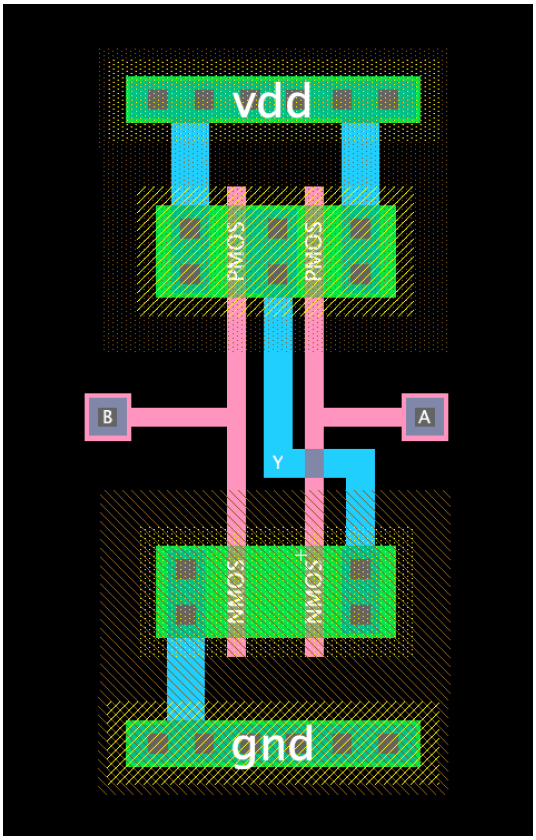
In this lab, we explored the design and simulation of basic CMOS logic gates and used them to build a full adder circuit. Starting with simple 2-input NAND and XOR gates, we created schematics, layouts, and simulations to understand how transistor sizing and routing affect logic behavior. After verifying these gates through DRC and NCC checks, we combined them to design a complete 1-bit full adder. The goal was to not only get functional circuits, but also to understand how signal timing, glitches, and layout organization influence digital circuit performance in real VLSI design.

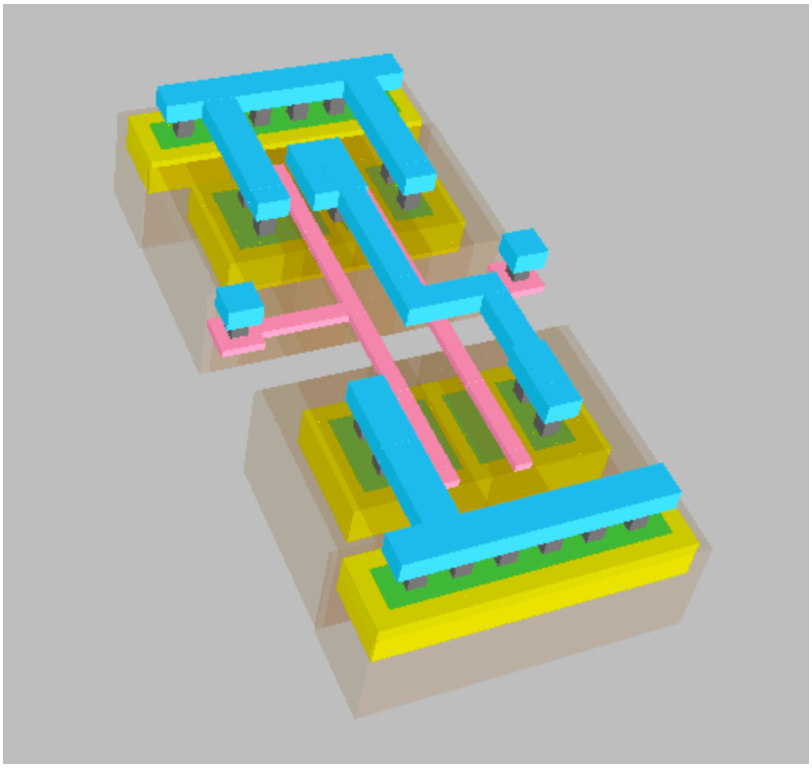
Part 1:

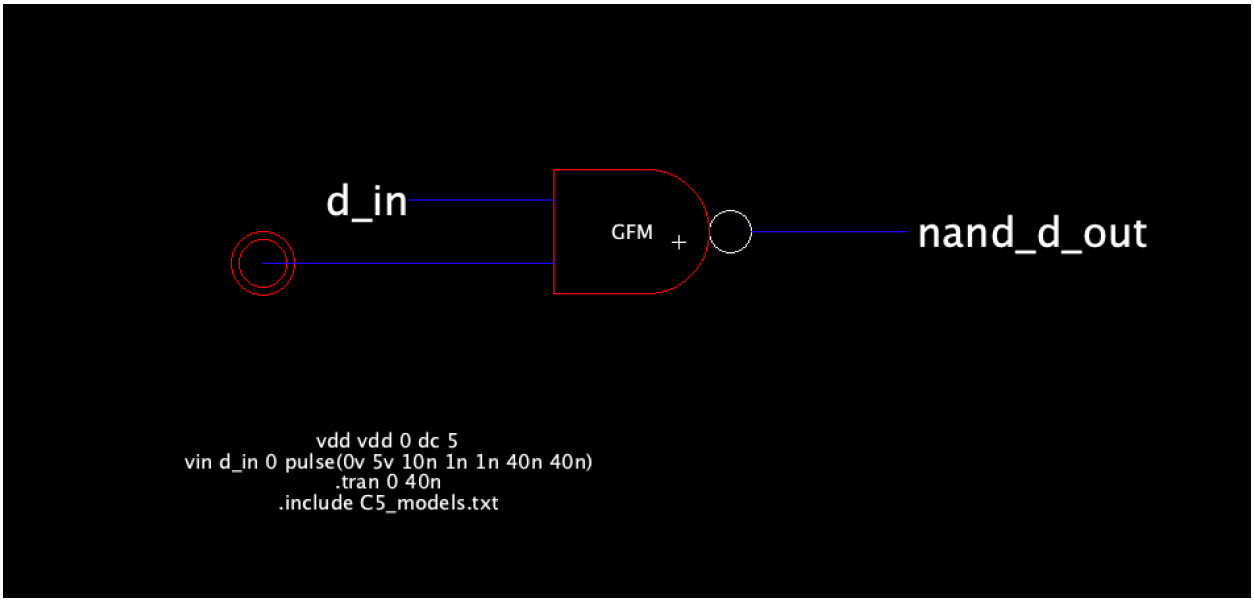
For this part of the lab, I drafted schematics, layouts, and symbols for a 2-input NAND gate and a 2-input XOR gate using 6 μm/2 μm NMOS and PMOS transistors. Each symbol followed standard logic gate conventions, with my initials placed inside the symbol for identification. All layouts were built within standard cell frames designed to snap together seamlessly for consistent VDD and GND routing. The inputs, outputs, and power lines were routed entirely in metal1 to maintain clear and uniform connectivity. Both gates were simulated for all four input combinations (00, 01, 10, 11) to verify proper logical behavior, and the designs successfully passed DRC (Design Rule Check) and NCC (Netlist Connectivity Check) to confirm their correctness and manufacturability.

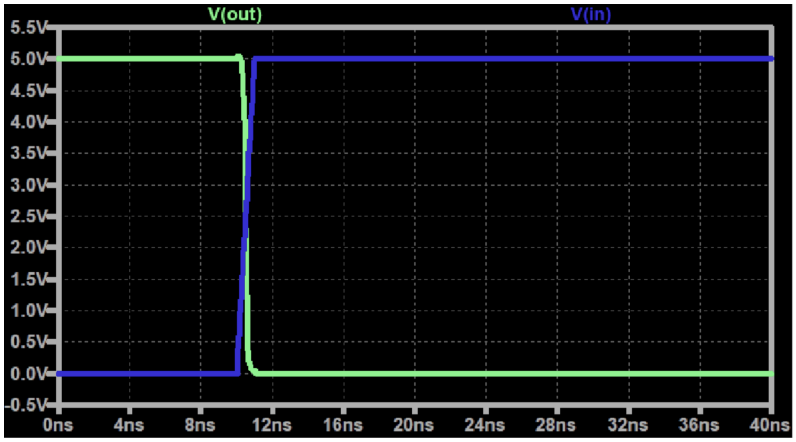
NAND gate:

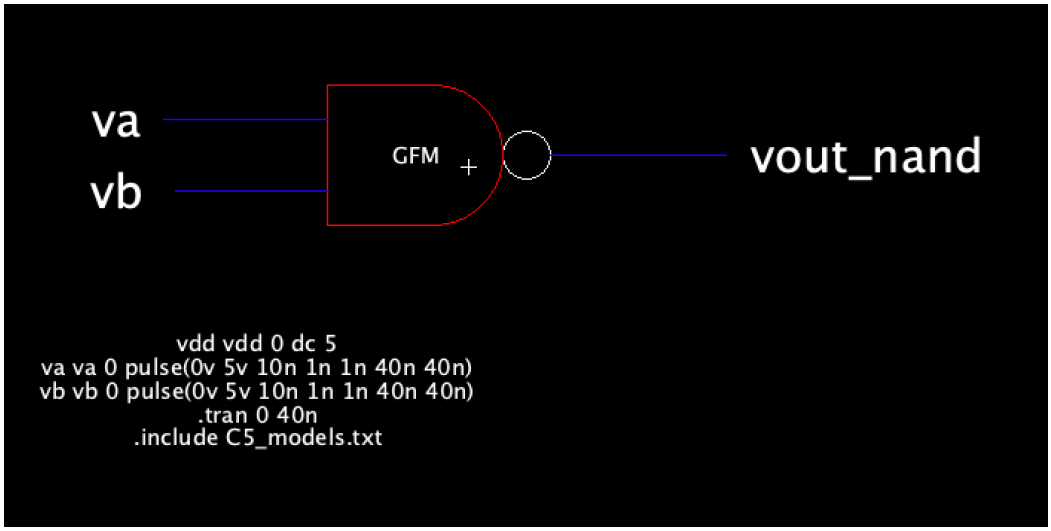


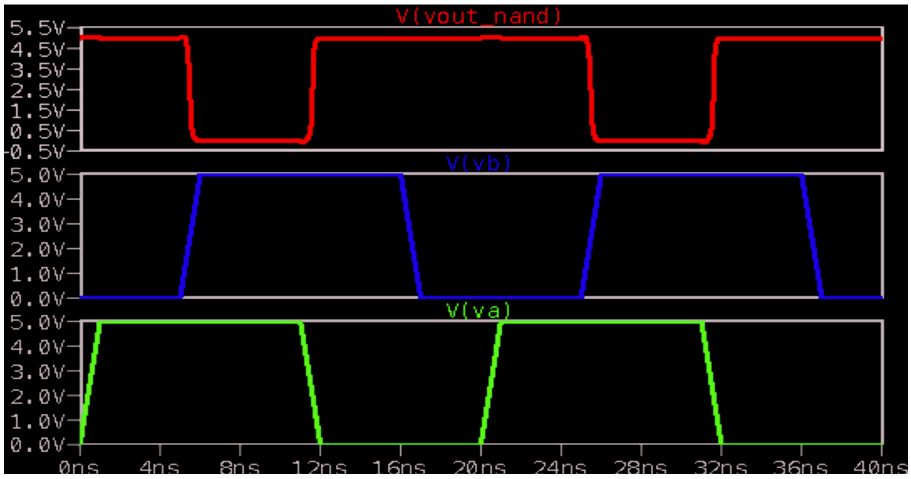

vdd
PMOS
PMOS
B
A
Y
NMOS
NMOS
gnd




d_in
GFM
nand_d_out
vdd vdd 0 dc 5
vin d_in 0 pulse(0v 5v 10n 1n 1n 40n 40n)
.tran 0 40n
.include C5_models.txt


V(out)
V(in)
5.5V
5.0V
4.5V
4.0V
3.5V
3.0V
2.5V
2.0V
1.5V
1.0V
0.5V
0.0V
-0.5V
0ns
4ns
8ns
12ns
16ns
20ns
24ns
28ns
32ns
36ns
40ns

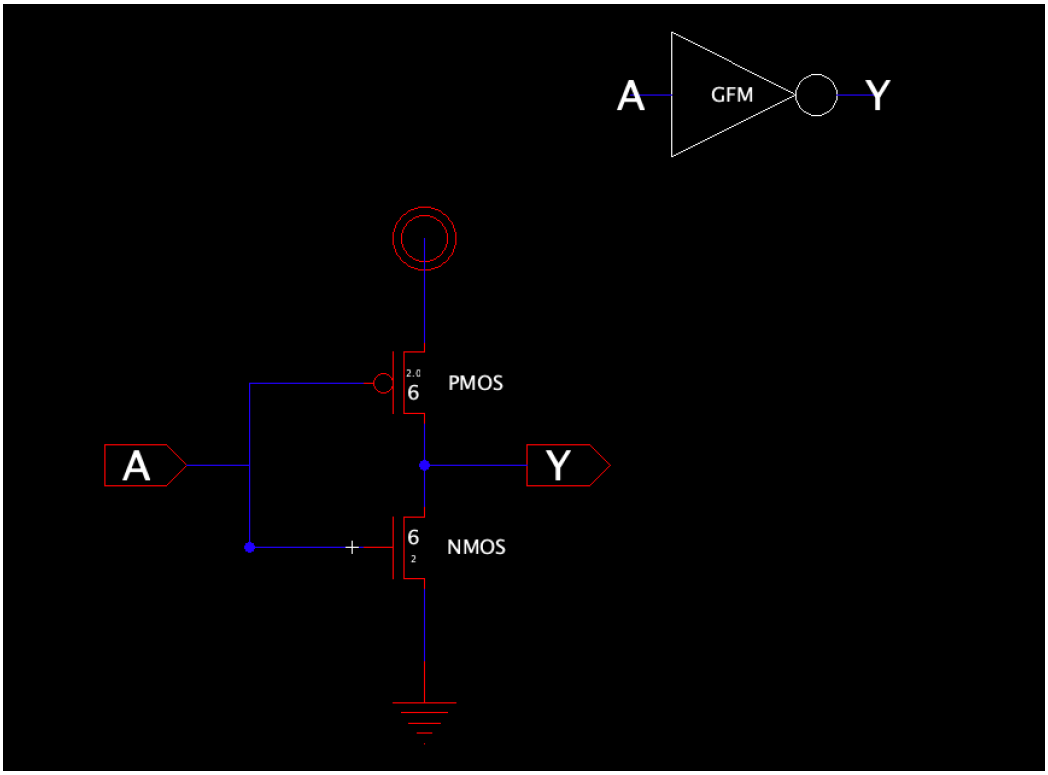


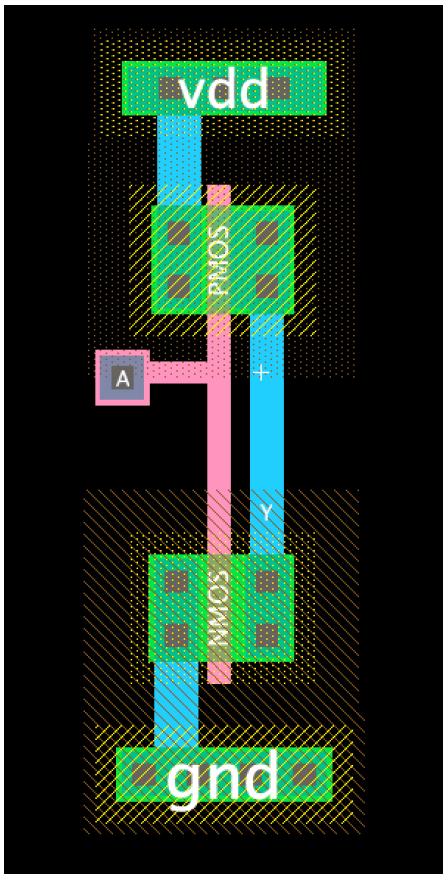


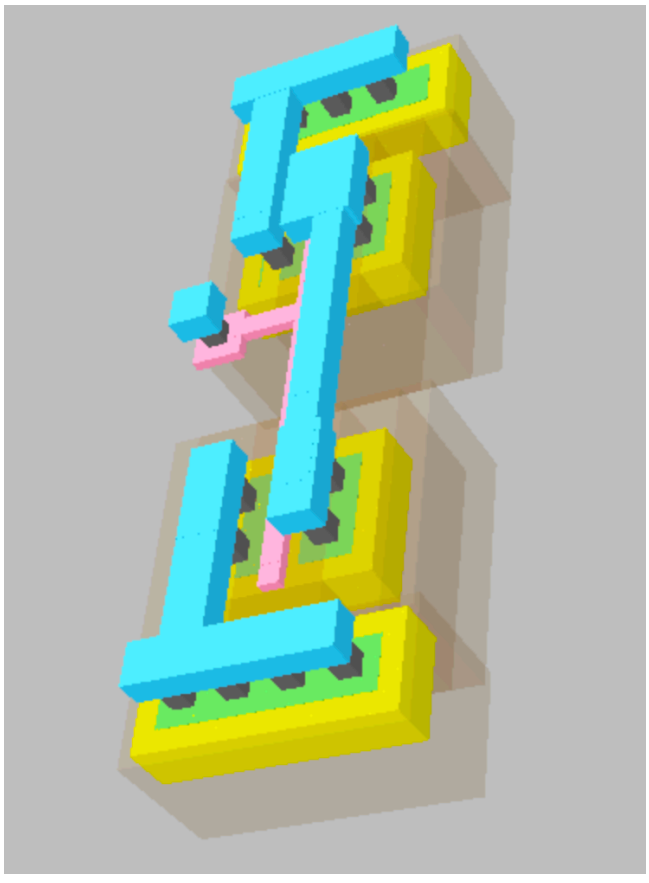
The 2-input NAND gate was designed using two PMOS transistors in parallel and two NMOS transistors in series, each with dimensions of 6 μm/2 μm. This configuration ensures that the output only goes low when both inputs are high, matching the expected NAND truth table behavior. The schematic shows clear routing of VDD and GND in metal1, while the layout follows the standard cell format for alignment and future connection in larger circuits. After completing the layout, both DRC and NCC checks were run to confirm there were no design or connectivity errors.

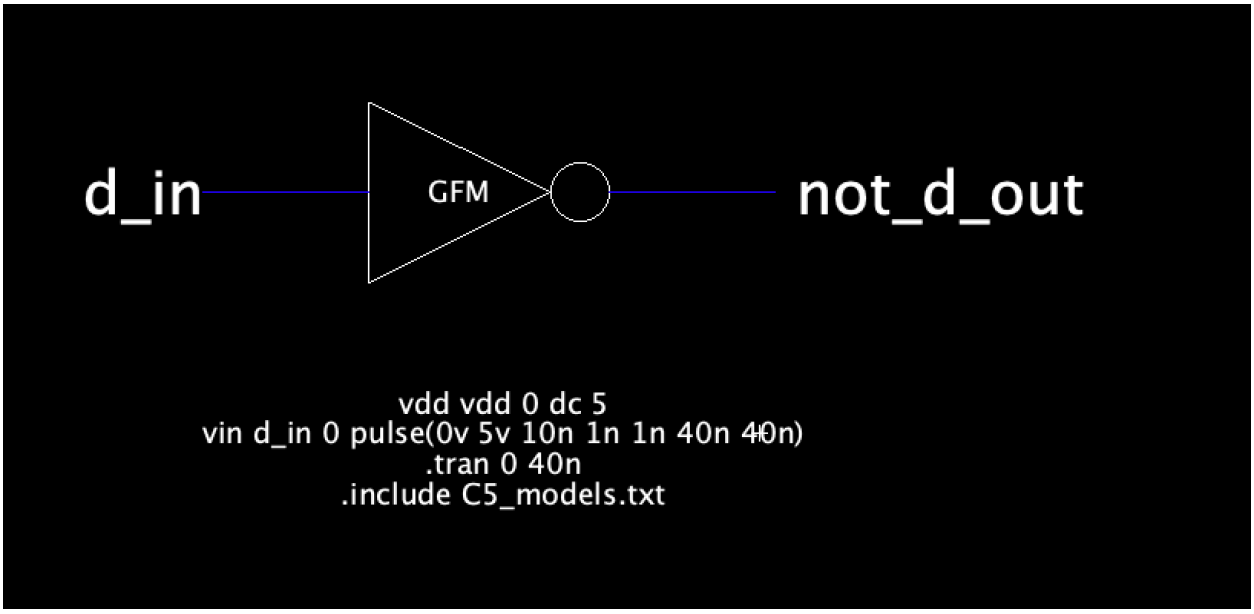
During simulation, the circuit was tested for all four possible input combinations, and the output followed the correct logic pattern. However, small glitches can occur in the output due to timing differences between the input signals. For example, if one input transitions slightly earlier than the other, the internal node connected to the series NMOS transistors may momentarily discharge before the second input switches. This short delay can cause a brief drop in the output voltage, even when the logical output should remain high. These glitches are common in real circuits and highlight the importance of matching input timing and considering propagation delays in VLSI design.

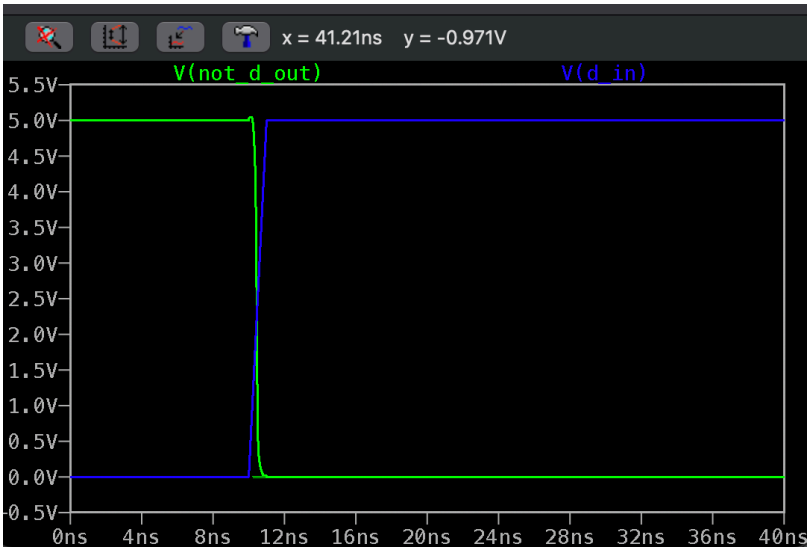
NOT gate:





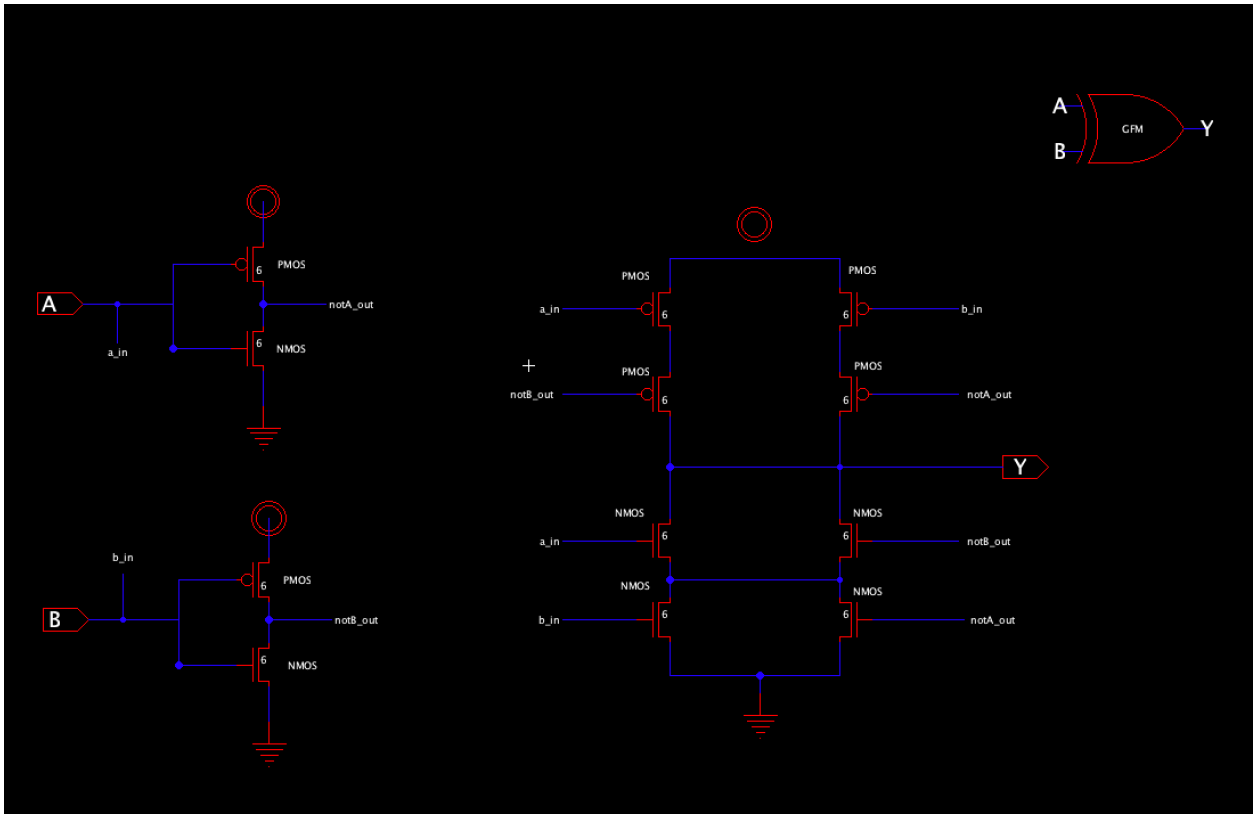


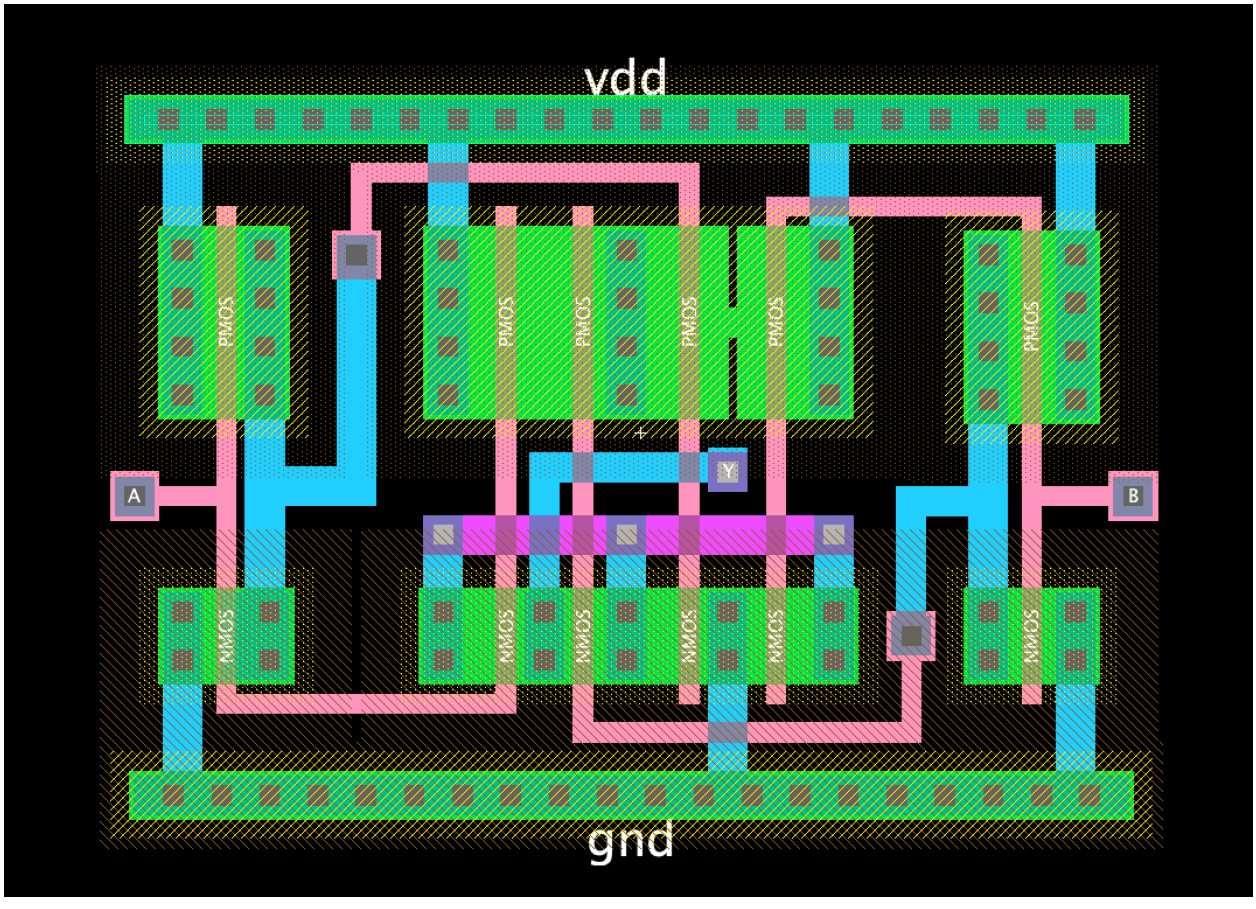


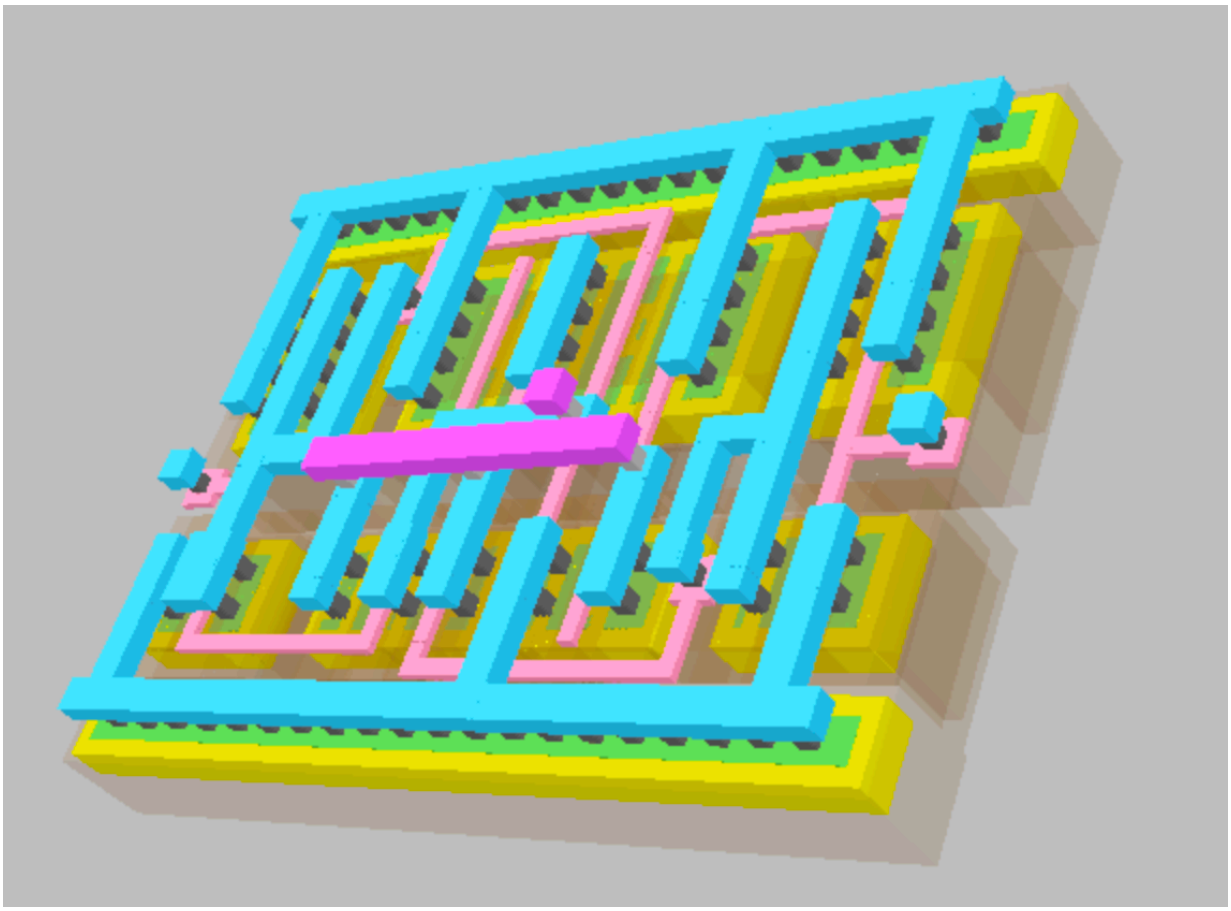


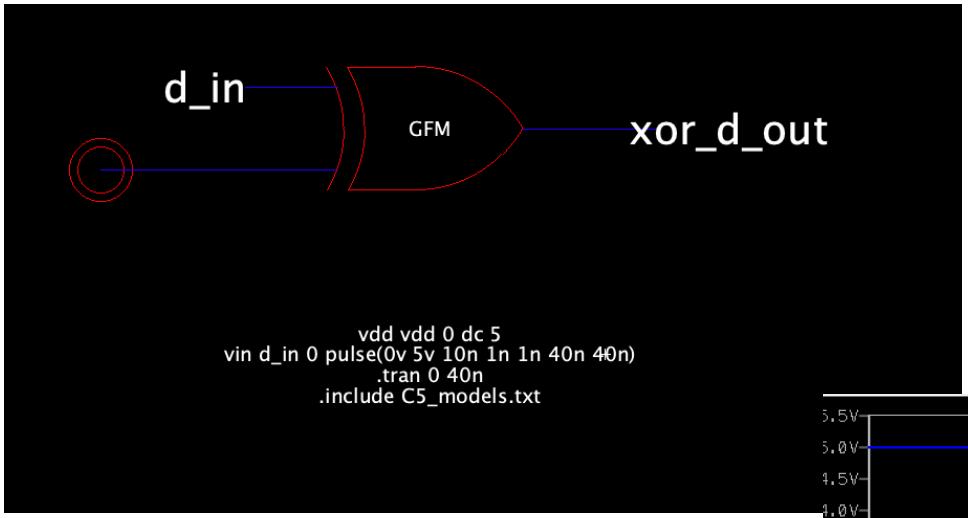
The NOT gate, or inverter, was built using one PMOS and one NMOS transistor, each with a width-to-length ratio of 6 μm/2 μm. The PMOS transistor connects between VDD and the output, while the NMOS connects between the output and ground, with both gates tied together to form a single input. This complementary configuration ensures that when the input is high, the output is pulled low, and vice versa. The schematic was implemented and then converted into a layout using metal1 routing for the input, output, VDD, and GND. Standard cell alignment was maintained for consistency with other logic gates in the lab. The circuit simulation confirmed the correct inverter behavior and revealed a brief transition delay between input and output changes. Although the NOT gate has only one input, timing mismatches or slow input transitions can still cause momentary glitches, typically small voltage fluctuations at the output, before the circuit fully settles. This emphasizes the impact of transistor switching speeds and parasitic capacitance in CMOS designs.

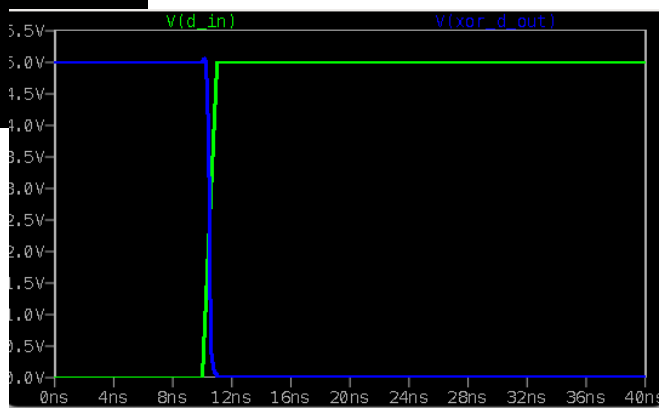
XOR gate:

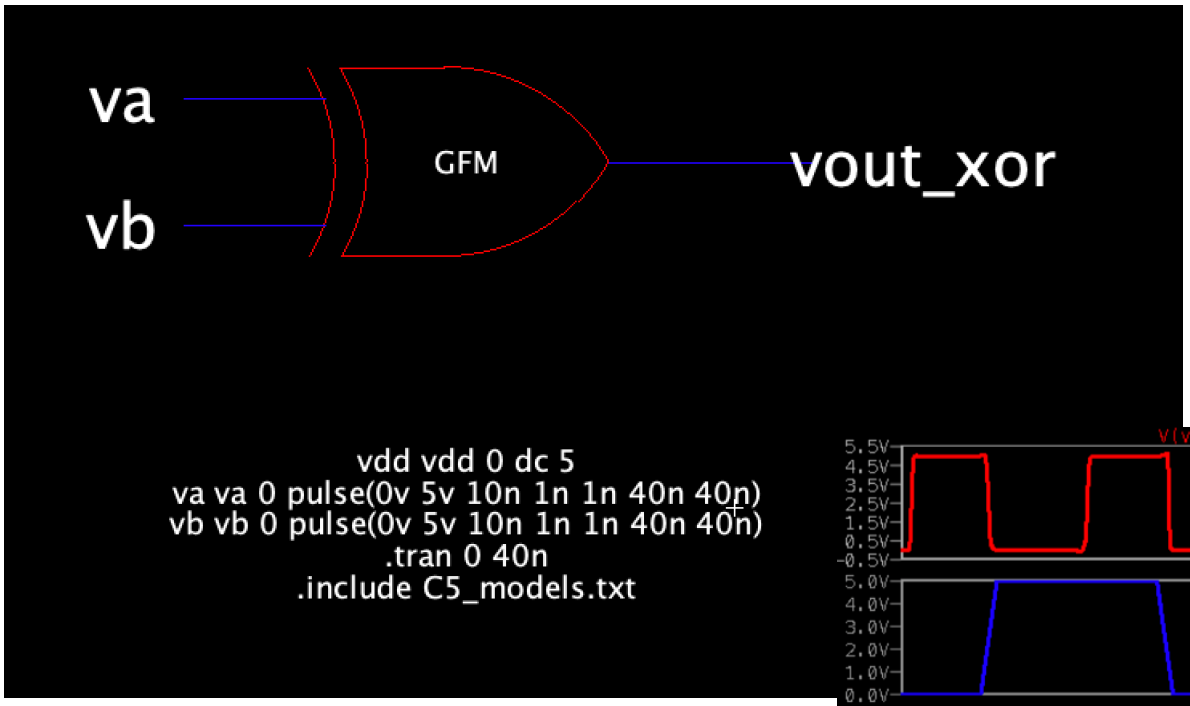

A
B
GFM
Y
PMOS
NMOS
notA_out
notB_out
a_in
b_in
6

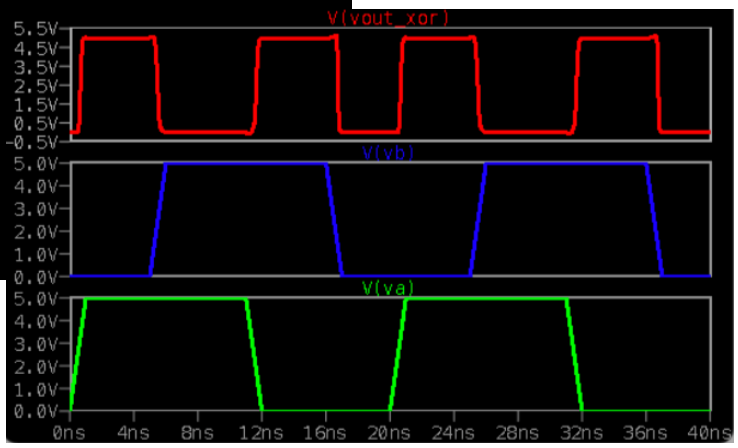

vdd
gnd
PMOS
NMOS
A
B
Y




d_in
GFM
xor_d_out
vdd vdd 0 dc 5
vin d_in 0 pulse(0v 5v 10n 1n 1n 40n 40n)
.tran 0 40n
.include C5_models.txt


V(d_in)
V(xor_d_out)
5.5V
5.0V
4.5V
4.0V
3.5V
3.0V
2.5V
2.0V
1.5V
1.0V
0.5V
0.0V
0ns
4ns
8ns
12ns
16ns
20ns
24ns
28ns
32ns
36ns
40ns



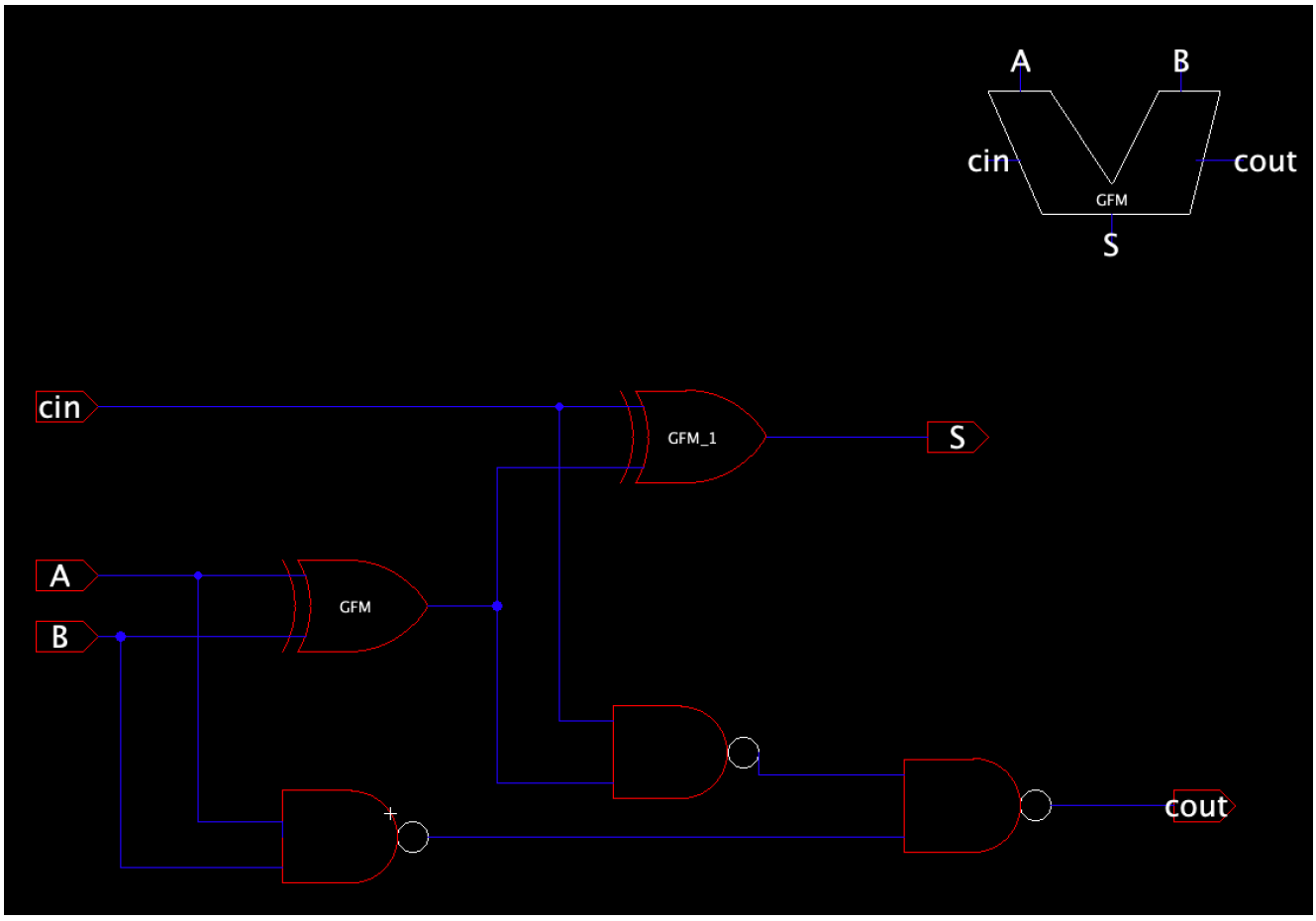


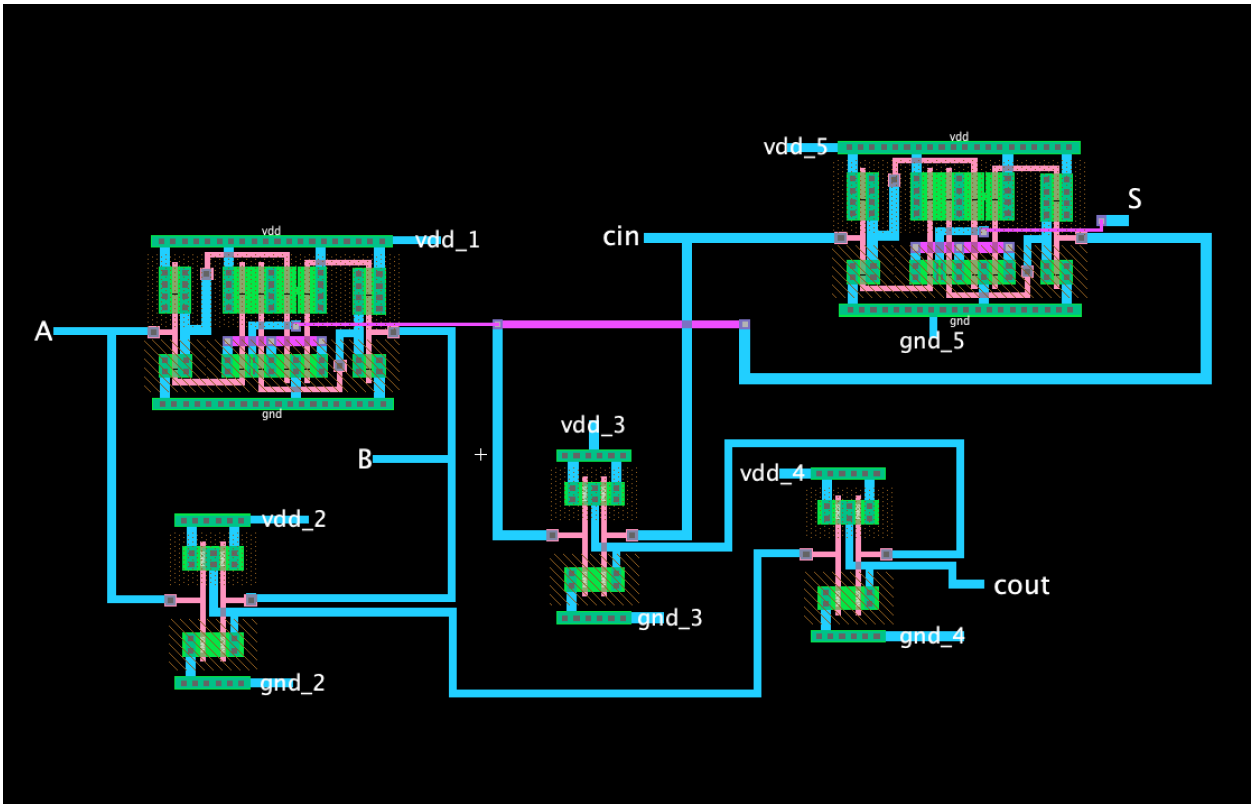
The XOR gate was designed to output a high signal only when the two inputs differ. Its schematic implementation uses 6 μm/2 μm NMOS and PMOS transistors in a complementary structure, combining multiple transistors to realize the Boolean expression $Y = A\overline{B} + \overline{A}B$. The design includes internal NOT gates to generate the inverted signals needed for this operation. In layout form, the transistors were arranged using a standard cell frame, with VDD and GND running horizontally and all routing completed in metal1 for inputs, outputs, and power connections. The simulation confirmed the expected XOR truth table, producing a high output only when inputs A and B were different (01 or 10). However, during transitions, small glitches can occur when input signals do not switch simultaneously. For example, if one input changes slightly before the other, intermediate states can temporarily cause both NMOS or PMOS networks to partially conduct, leading to short spikes or dips in the output voltage. These timing-related glitches highlight the sensitivity of XOR circuits to signal synchronization and propagation delay, especially in cascaded logic designs.

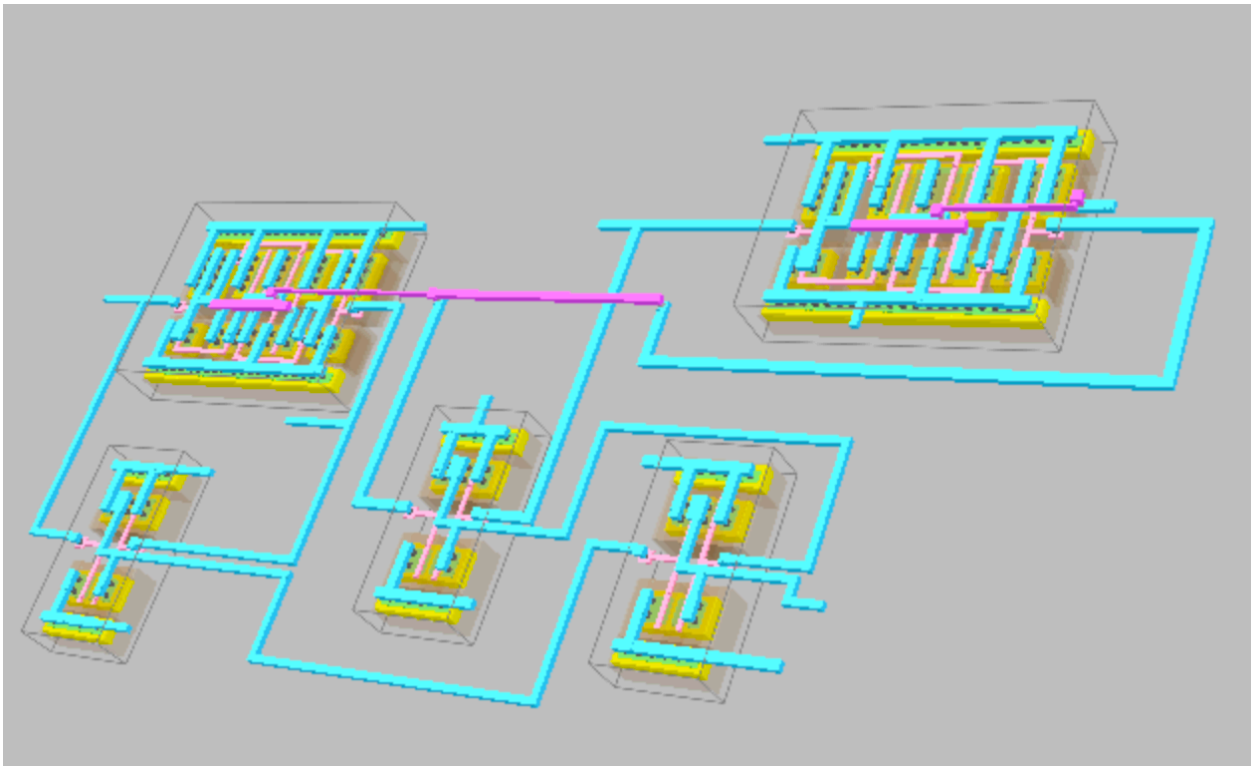
Part 2:

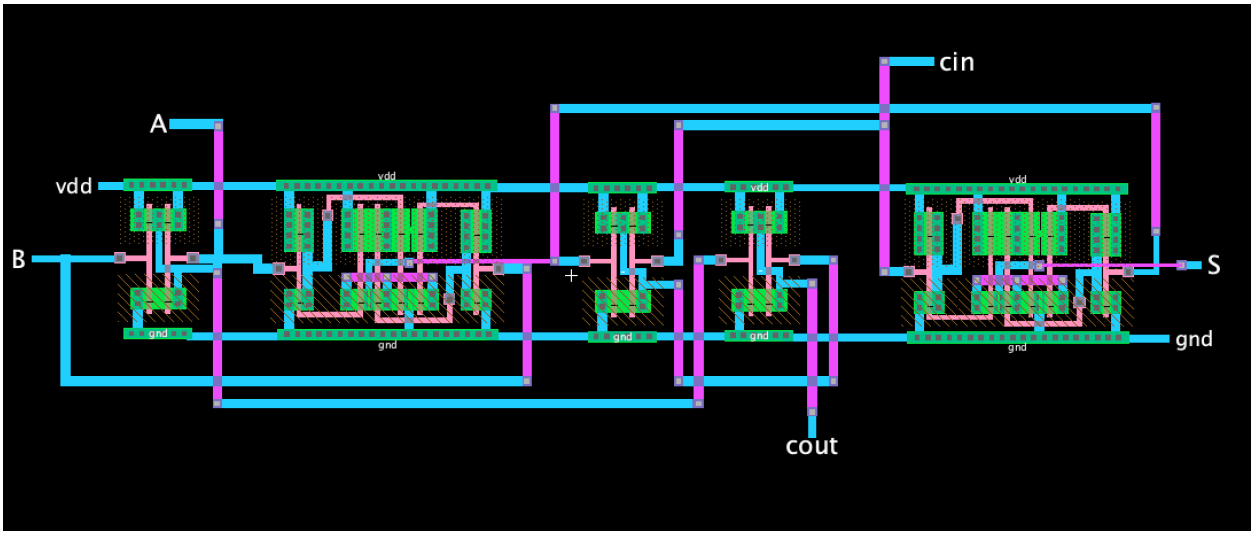
Using the verified NAND, NOT, and XOR gates from Part 1, I drafted the schematic, layout, and symbol for a 1-bit full adder that implements the standard truth table (Sum = A ⊕ B ⊕ Cin, Cout = majority(A, B, Cin)). The layout arranges the five required gates end-to-end as a standard cell row so VDD and GND run continuously across the top and bottom rails for clean power routing. Per the rules, I kept all internal gate routing in metal1 and placed only the primary I/Os (A, B, Cin, Sum, Cout) on metal2—avoiding any use of metal3. I then simulated the full adder across all eight input combinations to verify correct Sum and Cout behavior and to check for any timing-related hazards at the XOR stages. Finally, I completed DRC and NCC to confirm the design is rule-clean and that the extracted netlist matches the schematic connectivity.

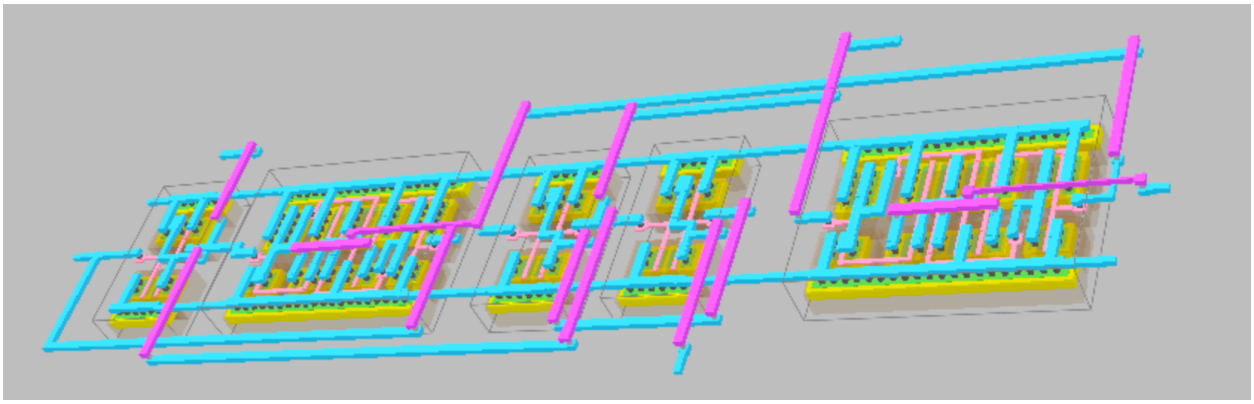
Full Adder:

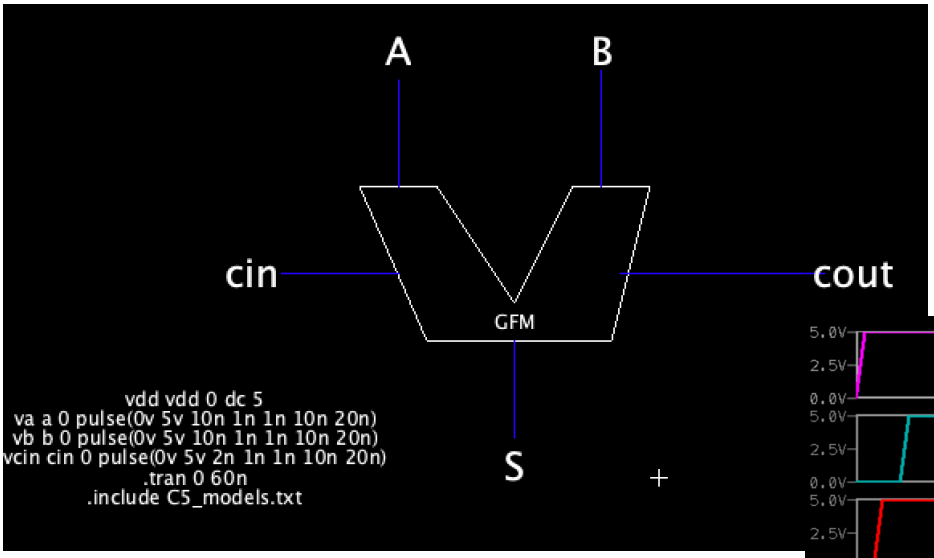


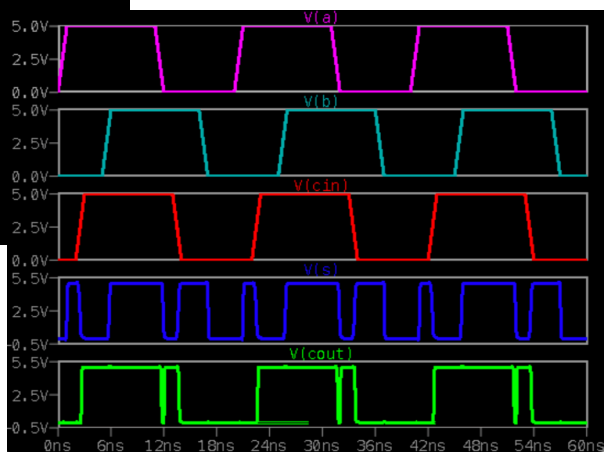

vdd_5
vdd
S
vdd
vdd_1
cin
A
gnd_5
gnd
vdd_3
B
vdd_4
vdd_2
cout
gnd_3
gnd_4
gnd_2




cin
A
vdd
B
S
gnd
cout




A
B
cin
cout
GFM
S
vdd vdd 0 dc 5
va a 0 pulse(0v 5v 10n 1n 1n 10n 20n)
vb b 0 pulse(0v 5v 10n 1n 1n 10n 20n)
vcin cin 0 pulse(0v 5v 2n 1n 1n 10n 20n)
.tran 0 60n
.include C5_models.txt


V(a)
V(b)
V(cin)
V(s)
V(cout)
5.0V
2.5V
0.0V
5.5V
-0.5V
0ns
6ns
12ns
18ns
24ns
30ns
36ns
42ns
48ns
54ns
60ns

The full adder circuit was implemented using the verified basic logic gates from Part 1, combining two XOR gates, two AND gates, and one OR (realized using NAND/NOT equivalents) to compute the standard Boolean expressions: Sum = A ⊕ B ⊕ Cin and Cout = AB + Cin(A ⊕ B). The schematic correctly reflected this logic, and simulation results matched the expected truth table across all eight input combinations, confirming accurate propagation of both the Sum and Carry outputs. During transient analysis, minor timing variations were observed due to different input pulse alignments, occasionally producing short glitches on Sum — a common occurrence caused by unequal path delays through the XOR stages. For the layout, the five gate blocks were placed end-to-end in a single row, maintaining continuous VDD and GND rails for clean power distribution. Metal1 was used for intra-cell routing, while inputs and outputs were routed in metal2 to comply with layer restrictions. The layout was verified with DRC (Design Rule Check) and NCC (Netlist Connectivity Check), both passing without violations, confirming electrical and physical correctness. A 3D layout visualization verified the layer stacking and interconnect alignment, completing the design of a fully functional CMOS full adder cell.

Conclusion:

To conclude, this lab provided valuable hands-on experience in designing, simulating, and laying out CMOS logic circuits from the ground up. Starting with basic gates like NAND, NOT, and XOR, and then combining them into a functional full adder, helped reinforce how transistor-level design translates into higher-level digital logic. The simulations confirmed correct logic behavior and highlighted important real-world considerations such as propagation delay and glitching due to input timing differences. Creating the layouts especially for the full adder was quite challenging, as it required careful alignment of standard cells, consistent power routing, and precise metal layer connections to ensure DRC and NCC compliance. Despite the difficulty, seeing the completed, functional layout made the process rewarding and deepened my understanding of CMOS design and layout methodology.